INSTITUTO FEDERAL DE SANTA CATARINA

ALESSANDRO MOSCHETTA
OTAVIO HENRIQUE MOSCHETTA

LINGUAGENS DE PROGRAMAÇÃO

Caçador - SC
05/09/2021

1 INTRODUÇÃO

A intenção inicial é conhecer quais linguagens de programação orientadas a objetos são usadas, a uma análise da literatura especializada no tema. Foi encontrada uma quantidade de linguagens de programação disponíveis, de naturezas e tipos de aplicação distintos. Abordamos com alguma profundidade algumas dessas linguagens.

Como dito por Roby (2018), do site de tecnologia Tech Republic, trabalhando com desenvolvimento de software, estamos constantemente em buscando da melhor linguagem de programação. Assim examinando a demanda do mercado, estamos avaliando de uma maneira a popularidade das linguagens de programação, e demonstra habilidades e aperfeiçoamento, mantendo assim um perfil profissional esperado pelas empresas, afirma Speros Misirlakis (2017) no seu blog Coding Dojo.

2 LINGUAGENS DE PROGRAMAÇÃO ORIENTADAS A OBJETOS

A programação orientada a objetos (POO) é uma ferramenta utilizada para o desenvolvimento de software com uma proximidade da realidade, seus componentes são representados no software, realizada através de objetos, como idos objetos da realidade, possuem características (atributos) e funções (métodos), nas quais são realizadas ações com os objetos mudando de estado. A POO traz algumas vantagens com relação à programação estruturada, tais como modularização, facilidade de extensão através de herança ou composição, facilidade de reutilização, entre outras. Para compreender POO é preciso ter em mente alguns conceitos.

2.1 Linguagem PHP

A linguagem PHP foi criada em 1994 por Rasmus Lerdof, um engenheiro de Software Canadiano-dinamarquês. Porém sua primeira versão foi criada para utilização pessoal, Lerdof pensou em criar algo para monitorar acessos ao seu site pessoal. Já em 1995, ele criou um pacote chamado Personal Home Page Tools pois estava adquirindo uma alta elevada de usuários do PHP. Sua segunda versão foi lançada com o título de PHP/FI, onde foi inserido o Form Interpreter, uma ferramenta analisadora sintaticamente com consultas de SQL.

Em 1997, aproximadamente 50000 sites em todo o mundo já contavam com uma pequena equipe de desenvolvimento que mantinha o projeto com contribuições de desenvolvedores e usuários em todo o mundo. A partir da versão 3, dois programadores israelenses, Zeev Suuraski e Andi Gutmans reescreveram completamente a arquitetura da linguagem, porém os objetos ainda não eram considerados uma parte necessária da nova sintaxe, embora tenha sido adicionado, já nesta versão, um pequeno suporte a classes.

Em 1998 haviam mais de 100000 domínios do PHP, ultrapassaria um milhão de domínios um ano mais tarde. Para a versão 4, o mecanismo Zend (nome derivado de Zeeve e Andi), que movia a linguagem, foi escrito do zero, segundo Zandstra (2006), porém deixando algumas falhas, como a atribuição de objetos à variáveis, que ainda se dava por valores e não por referência. Assim cada atribuição de um objeto à uma variável, a passagem do mesmo a uma função ou o retorno desta causava a cópia do objeto.

Segundo Zandstra (2006), o PHP 5 representa, explicitamente, o endosso aos objetos e à programação orientada a objetos. O PHP continua não sendo uma linguagem orientada a objetos, porém já permite a utilização do pleno potencial da orientação a objetos.

Em 2000 com o lançamento da versão 4.0, trazendo melhorias a sessões, suporte a vários tipos de servidores e permitindo ser utilizado como linguagem de shell script, mas ainda a linguagem necessitava de um suporte robusto para o paradigma de orientação a objetos, isso ocorreu em 2004 com o lançamento da versão 5.0, que possibilitou o desenvolvimento de projetos corporativos utilizando o paradigma e o padrão Model-View-Controller (DALL'OGLIO, 2015).

Antes da abordagem orientada a objetos ser popularizada, nos períodos de 1975 e 1985, foram feitas grandes soluções com o desenvolvimento estruturado, estas eram compostas por análise estrutural, análise de fluxo de dados, programação e testes estruturados, portanto com o passar do tempo novas tecnologias e necessidades corporativas surgiram e esse tipo de abordagem clássica começou a enfrentar problemas no desenvolvimento de software. Por muitas vezes as técnicas estruturais eram incapazes de lidar com sistemas de grande porte que de fato geravam códigos desorganizados, ou seja, as técnicas foram criadas para suportar sistemas menores com uma estimativa de 5.000 linhas de código ou de médio porte com 50.000 linhas. Outro ponto fundamental do problema era o gasto de recursos financeiros e o tempo com manutenção do sistema após ser entregue, no qual chegava a ser cerca de 70% (SCHACH, 2011).

O paradigma de programação orientada a objetos é uma abordagem para elaboração de sistemas, que envolve desde a modelagem do sistema até seu desenvolvimento utilizando objetos que se relacionam (DALL'OGLIO, 2015), é um modelo de objeto engloba princípios da abstração, encapsulamento, modularidade e herança (BOOCH, 1994), os conceitos começaram a ganhar notabilidade durante as décadas de 1980 e 1990 quando a programação estrutural ainda era dominante entre os desenvolvedores. De acordo com o autor Schach (2011, p. 20), o paradigma aplicado de forma correta incentiva a prática de reutilização de código, simplificando o desenvolvimento e a manutenibilidade de um sistema.

> O paradigma de orientação a objetos reduz o nível de complexidade de um produto de software e, portanto, simplifica tanto o desenvolvimento quanto a manutenção. O paradigma de orientação a

> objetos permite reutilização; pelo fato de os objetos serem entidades independentes, eles podem ser utilizados em novos produtos futuros. Essa reutilização de objetos reduz o tempo e o custo, tanto em termos de desenvolvimento quando de manutenção. (SCHACH, 2011, p. 20)

2.2 Linguagem C e C++

De acordo com Schildt (1996), quem criou a linguagem C foi Dennis Ritchie em 1972 nos laboratórios Bell, tinha como propósito utiliza-la no sistema operacional Unix, mas a linguagem foi utilizada por muitos sistemas operacionais.

Como ela tem um fácil acesso ao hardware, e a programação direta em microprocessadores, requisitos de memória baixos, uma rapidez obtida na linguagem de máquina, é a linguagem de mais baixo nível, possibilitando assim a implantação de programas em Assembly, esta é a linguagem de máquina, permitindo assim resolver problemas em que a dependência do tempo é crucial. Mesmo não sendo tão portável e prático como Java, é possível o envio dos códigos de uma máquina para outra, quase nenhuma alteração. Isso se dá devido ao padrão American National Standards Institute (ANSI) estabelecido na linguagem, que define as características dos compiladores.

A linguagem de programação C é de alto nível, propriedade essa que deixa subentendido que parte da função abstrai a complexidade das linguagens de máquina. Para o Java é diferente, C é uma linguagem compilada, seus códigos são transformados diretamente em linguagem de máquina, por isso sempre utiliza outro sistema operacional, o programa é recompilado e pode ser necessária alguma alteração (BACKES, 2013).

Já a linguagem C é denominada por Damas (1999) como de *general purpose*, não tendo uma área de desenvolvimento específica, a Common Business Oriented Language (COBOL), para processamento de registros, e Formula Translation (FORTRAN), utilizada para cálculos científicos. Mas tendo como vantagem, pode se adequar para qualquer tipo de projeto, desde processamento de registros, interfaces gráficas e até mesmo alguns sistemas operacionais utilizam.

Já para o C++ uma linguagem que deriva de C, seus incrementos têm como objetivo o suporte à programação orientada a objetos. C++ utiliza a mesma sintaxe da linguagem C, com pequenas exceções os programas de C também são programas de C++. Grandes programas como Adobe Photoshop, Mozilla Firefox,

Internet Explorer e partes do Microsoft Windows foram desenvolvidos em C e C++.

2.3 Linguagem SCALA

Criada em 2001 na École Polytechnique Fédérale de Lausanne (EPFL), Lausana na Suíça, por Martin Odersky (ex-integrante do Java Generics), a linguagem Scala liberada publicamente na plataforma Java em janeiro de 2004. Já a versão alternativa para .NET foi publicada só em junho. Somente em março de 2006, foi liberada a segunda versão da linguagem (ODERSKY, 2014, p.01).

Scala é a mistura de “scalable” e “language”, nome escolhido para a linguagem mais suscetível a substituir o tão forte Java. Scala é utilizada tanto para escrita de pequenos scripts até construção de grandes sistemas.

É considerada uma linguagem multiparadigma, com métodos de programação funcional orientada a objetos, roda em três diferentes plataformas: Android, JVM (Java Virtual Machine), e NET (ODERSKY; SPOON; VENNERS, 2008, p.39).

Após ser compilado, o código Scala gera Java bytecodes, sendo possível a utilização do Java sem a necessidade de interfaces ou cópia de código (ODERSKY; SPOON; VENNERS, 2008, p.39).

Ótima linguagem para criar scripts com vários componentes Java, sua força é mais notável no emprego de grandes sistemas e frameworks onde são reutilizáveis alguns componentes. Scala é a mistura de orientação a objetos e programação funcional em uma linguagem tipada.

Essa fusão pode ser observada em diferentes aspectos, mais quando fala-se em escalabilidade, o funcional da linguagem é fácil e rápido nas construções simples, a parte orientada a objeto possibilita adaptação de grandes demandas. Tornando possível a criação de padrões de programação e abstração de componentes, também um estilo legível e mais conciso de programação, torna a programação em Scala divertida (ODERSKY; SPOON; VENNERS, 2008, p.39).

Eric Raymond (1999 apud ODERSKY; SPOON; VENNERS, 2008, p. 41) diz que podemos usar uma catedral e um bazar como duas metáforas para construção de softwares.

> The cathedral is a near-perfect building that takes a long time to build. Once built, it stays unchanged for a long time. The bazaar, by contrast, is adapted and extended each day by the people working in it.

Esta linguagem não oferece tudo o que o usuário necessita, ao invés ela disponibiliza as ferramentas necessárias para que ele crie da forma que ele precisar (ODERSKY; SPOON; VENNERS, 2008, p.41).

Uma alternativa segura é proposta em Java, a arquitetura baseada em trocas de mensagens, parecida com o modelo usado pela linguagem Earlang. Java possui uma biblioteca baseada em threads, pode ser usada por programadores Scala, como qualquer outra biblioteca Java, Scala conta com biblioteca adicional que implementa o modelo de atores adotado na linguagem Earlang (ODERSKY; SPOON; VENNERS, 2008, p.42-45).

3 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo fazer um levantamento de algumas linguagens de programação orientadas a objeto, mais utilizados no mercado de trabalho. Para tanto, valeu-se de uma pesquisa bibliográfica sobre algumas linguagens do mercado global, uma análise documental. O resultado geral foi de que as linguagens estão buscando atualizações constante, uma vez que o mercado está exigindo ideias novas com muita frequência. Contudo, mostra que a linguagem Java, ocasionalmente identificada como a mais popular no mercado, está ganhando um bom espaço no mercado atual.

REFERÊNCIAS

BACKES, A. Linguagem C completa e descomplicada. Rio de Janeiro: Elsevier, 2013.

BOOCH, G. Object-oriented analysis and design with applications. 2 ed. Santa Clara: Addison-Wesley, 1994.

DALL'OGLIO, P. PHP: programando com orientação a objetos. 3 ed. São Paulo: Novatec, 2015.

DAMAS, L. Linguagem C. Lisboa: FCA, Editora de Informática, 1999.

HORSTMANN, Cay S. Scala for the Impatient. 2012. Addison-Wesley Professional; 1 edition – 384 p.

MISIRLAKIS S. The 7 most in-demand programming languages of 2018. 2017. Disponível em: https://www.codingdojo.com/blog/7-most-in-demand-programming languages-of-2018 Acesso em: 7 mar 2019.

ODERSKY, Martin. Scala by Example.École Polytechinique Fédérale de Lausanne – EPFL – Suíça, 2013 – 137 p.

ODERSKY, Martin. The Scala Language Especification. École Polytechinique Fédérale de Lausanne – EPFL – Suíça, 2014 – 183 p.

ODERSKY, Martin; SPOON, Lex; VENNERS, Bill. Programming in Scala.2008.Artima Press: Califórnia, 2008 – 754 p.

ROBY, K. The top 3 programming languages developers actually use. Tech Republic. 2018. (02m13). Disponível em: https://www.techrepublic.com/videos/the-top-3-programming-languages-developers-actually-use/ Acesso em: 07 jun 2019.

SCHACH, S.R. Object-oriented and classical software engineering. 8 ed. New York: McGraw-Hill, 2011.

SCHILDT, H. Turbo C: guia do usuário. São Paulo: Pearson Education/Makron Books, 1994.

ZANDSTRA. Matt. Entendendo e Dominando o PHP. 1. ed. São Paulo: Digerati Books, 2006.